AssemblyBench: Physics-Aware Assembly of Complex Industrial Objects –Supplementary Materials–

Danrui Li[1*] Jiahao Zhang[2*] Bernhard Egger[3]
Moitreya Chatterjee[4] Suhas Lohit[4] Tim K. Marks[4] Anoop Cherian[4]
[1]Rutgers, The State University of New Jersey, USA [2]The Australian National University, Australia
[3]Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany [4]Mitsubishi Electric Research Laboratories (MERL), USA
[1]danrui.li@rutgers.edu [2]jiahao.zhang@anu.edu.au [3]bernhard.egger@fau.de [4]{chatterjee,slohit,tmarks,cherian}@merl.com

https://www.merl.com/research/highlights/assemblybench

Table of Contents

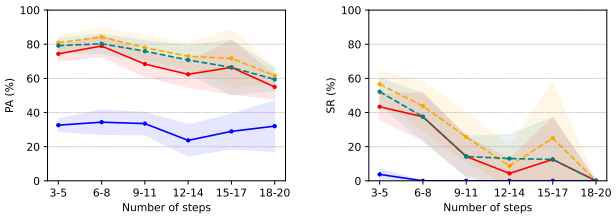
1. Detailed Performance Analysis

1.1. Effect of the Number of Steps

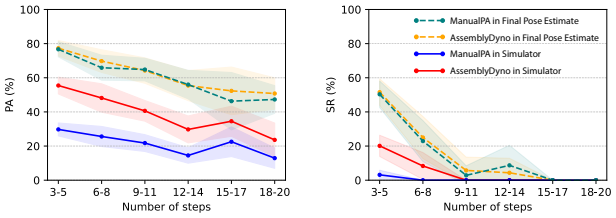
As shown in Figure 1, across both settings (GT order and Standard) and both metrics (PA, SR), all curves decline as the number-of-step bin increases, indicating that longer sequences, which contain more complex diagrams and assembled geometries, are harder. SR drops more steeply than PA and often approaches zero for long sequences under simulation, acting as the most strict metric in our study.

Meanwhile, it shows AssemblyDyno is more robust in simulation. In the simulation protocol (solid lines), *AssemblyDyno* (red) consistently lies above *ManualPA* (blue) for both PA and SR across nearly all number-of-step bins and in both settings, showing stronger execution robustness.

Again, the figure demonstrates our simulation is the stricter evaluation. For every method, metric, and setting, solid lines are lower than dashed lines (final-pose evaluation), confirming that simulation reveals failures that static end-state checks miss.



(a) GT order setting



(b) Standard setting

Figure 1. **Performance as a function of number of steps**. We present the PA and SR metric in two evaluation protocols (static final pose and simulation), for AssemblyDyno and ManualPA in two experiment settings. Shaded areas represent 95% confidence intervals of the metric.

*Work done during internships at MERL.

1.2. Effect of Trajectory Category

We compare our work against the ManualPA baseline [3] across multiple trajectory categories and two evaluation settings. As shown in Table 1, both approaches perform well on stationary trajectories. Stationary trajectories are always the first step of the assembly, where both the context and motion are simple. In contrast, for complex motions such as rotational or insert-and-rotate trajectories, the performance of both methods drops, but our approach remains substantially more robust, achieving noticeably higher PA and lower geometric error. Overall, across most categories and in both settings, our method outperforms ManualPA, doubling the improvement in PA.

Table 1. **Comparison of assembly performance across trajectory categories.** For each category, we report three metrics computed using our simulation-based evaluation protocol: median Average Chamfer Distance (mACD), median Final Chamfer Distance (mFCD), and Percentage of Accurate assemblies (PA). We compare them against corresponding results from the baseline (*ManualPA*[3]).

Category	**mACD** (10^{-3})↓		**mFCD** (10^{-3})↓		**PA (%)**↑	
	Ours	[3]	Ours	[3]	Ours	[3]
Standard Setting						
All	**34.82**	192.79	**15.97**	56.04	**42.9**	23.2
Stationary	**5.92**	108.21	5.72	**4.41**	61.1	**62.9**
Translational	**33.73**	187.06	**15.36**	50.92	**43.8**	24.5
Rotational	**50.48**	272.77	**26.76**	159.05	**30.6**	5.6
Insertion	**147.83**	250.08	**10.15**	40.26	**48.8**	26.7
Insert + Rotate	206.59	**189.52**	**44.20**	82.64	**14.3**	0.0
GT Order Setting						
All	**9.97**	230.52	**4.43**	35.00	**70.1**	31.3
Stationary	**2.78**	120.48	2.72	**2.53**	**77.5**	76.4
Translational	**9.43**	224.32	**4.30**	30.25	**70.9**	33.1
Rotational	**14.49**	373.30	**7.21**	215.70	**59.7**	6.5
Insertion	**100.24**	288.18	**5.92**	40.24	**73.3**	34.9
Insert + Rotate	**117.38**	166.57	**10.36**	67.83	**42.9**	0.0

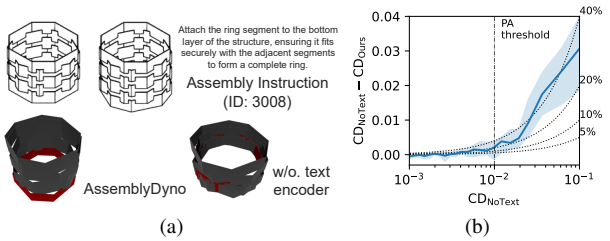
1.3. Effect of Loss Design

Ablation Study. From the ablation results in Table 2, we observe that all loss components in our framework are essential. Removing any individual loss term ($\mathcal{L}_P$, $\mathcal{L}_T$, $\mathcal{L}_R$, or $\mathcal{L}_{SR}$) consistently degrades performance across both the *Final Pose Estimate* metrics and the *Assembly in Simulator* metrics. The drops are particularly notable in the simulation-based metrics (mACD, mFCD, PA, and SR), indicating that each loss contributes critically to enabling the model to produce physically executable assembly trajectories. These observations confirm the necessity of the full loss design used in AssemblyDyno.

Sensitivity Analysis We further conduct a sensitivity analysis by varying the weights of the rotational loss λ_R and the rotation-regularization loss λ_{SR}. The original weights are in Table 4. Across the tested weight settings, the resulting performance metrics exhibit only small fluctuations. This low variance indicates that our method is robust to moderate perturbations of these hyperparameters. Thus, within the examined range, the overall assembly performance is not highly sensitive to the specific weight choices of these losses, demonstrating stability of the training objective.

1.4. Effect of Text Instructions

While the aggregated performance gains in Table 2 may appear marginal (+2%), the influence of text is not sufficiently unveiled. Specifically, for challenging assemblies such as the one in Fig. 2a, we find that incorporation of text leads to significant benefits. In Fig. 2b, we plot the median improvement in part-wise chamfer distance (CD) by AssemblyDyno, as a function of CD of the text-omitted version. Clearly, text instructions yield significant gains (up to 40%) for the worst cases ($\text{CD} > 10^{-2}$, the right half of the plot). This is currently not reflected in the PA metric we report, as it only counts the fraction of parts with $\text{CD} < 10^{-2}$. We will include this in the final paper.



(a) (b)

Figure 2. Text helps for difficult cases. **(a)** Example instruction step (top) and resulting prediction with vs. without using text (bottom). **(b)** Median value of $\text{CD}_{\text{NoText}} - \text{CD}_{\text{Ours}}$, plotted vs. $\text{CD}_{\text{NoText}}$.

1.5. Adding Multiple Parts in One Step

To evaluate the robustness of our model when multiple parts are added in a single step, we trained and tested our model while randomly removing (masking) the diagrams for up to 2 steps from each assembly. Table 3 shows that our model is significantly more robust to missing step diagrams than the Manual-PA baseline.

2. Physics-aware or physics-in-the-loop?

In this paper, we use the term "physics-aware" to indicate that a physics engine is used to generate the ground-truth part trajectories in our dataset and to evaluate the assemblies predicted by models.

We attempted a "physics-in-the-loop" strategy for our model, where physics constraint signals are included in the training loss. However, we found it to be less effective than

Table 2. **Part assembly results on the test split of AssemblyBench**. Best results are in **bold**, second best in blue.

Model	Final Pose Estimate			Assembly in Simulator			
	SCD(10^{-3})↓	PA(%)↑	SR(%)↑	mACD(10^{-3})↓	mFCD(10^{-3})↓	PA(%)↑	SR(%)↑
AssemblyDyno	3.87	**79.69**	44.29	9.97	4.43	70.15	**33.57**
w/o $\mathcal{L}_P$	4.09	77.75	38.57	10.79	5.11	67.91	27.50
w/o $\mathcal{L}_T$	4.19	68.43	27.86	17.69	6.92	57.41	25.00
w/o $\mathcal{L}_R$	3.97	73.71	37.14	11.94	5.26	64.57	26.43
w/o $\mathcal{L}_{S_R}$	3.85	79.17	43.21	16.98	11.06	47.18	5.36
Sensitivity Analysis							
$\lambda_R = 1$	3.88	78.64	42.86	10.15	4.59	69.11	30.00
$\lambda_R = 10$	3.74	79.49	**46.43**	**9.16**	**4.24**	**70.20**	32.14
$\lambda_{S_R} = 1$	**3.71**	79.61	44.64	10.93	5.65	64.35	20.00
$\lambda_{S_R} = 10$	3.82	79.20	40.71	10.04	**4.24**	70.04	30.36

Table 3. When step diagrams are randomly masked out (keeping text), AssemblyDyno is much more robust than ManualPA.

Model	Masked?	SCD(10^{-3})↓	PA(%)↑	SR(%)↑
AssemblyDyno	-	3.81	77.08	40.36
	✓	**4.61**	**69.20**	**22.86**
ManualPA	-	4.15	77.40	39.28
	✓	6.02	60.90	9.286

supervised training using ground-truth assembly trajectories.

This claim is supported by experiments where we use simulator refinements at test time: when predicted final part poses exhibit even minor interpenetrations (on the order of mm), we use a physics simulator to resolve these by pushing parts apart, resulting in large displacements from the ground-truth poses (see Figure 3 left). Such refinements is detrimental, reducing PA from 79.69% to 70.53% and SR from 44.29% to 36.79%.

In Figure 3 right, we conceptually illustrate this difficulty using a hypothetical ground-truth part trajectory and the directions of collision-induced forces from the simulator. As shown, when ground-truth trajectories are available, they can provide substantially stronger and more stable learning signals than the noisy collision gradients from the simulator. Consequently, designing effective collision-aware losses or post-processing schemes would require significant innovations beyond the scope of this work.

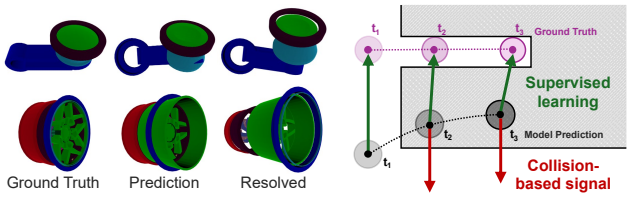


Figure 3. **Left**: Physics-based post-processing of an assembly. **Right**: Supervised learning guides predictions towards ground truth, while collision forces push parts to the nearest free space.

3. Physics Simulator Configurations

We present the most important simulator configurations in Table 5, which consists of general settings such as gravity and simulation substeps, as well as material settings that determined the friction behaviors of the shape. The friction parameters `kf` and `mu` need to be set to 0 in our evaluation. as shown in Figure 4, when friction effects are not disabled, some insertion behaviors will not be executed, even if we use ground truth trajectories to guide the simulator.

While we use a specific simulator [1] in our study, our evaluation protocol (*i.e.* the simulation design and its metrics) is agnostic to simulator choice, as long as it supports collision detections on non-convex mesh geometries.

Table 4. Hyperparameters for training AssemblyDyno.

Name	Value	Name	Value
Batch size	64	Epoch	1,000
Optimizer	AdamW	Learning rate	4×10^{-5}
Weight decay	1×10^{-4}	Betas for AdamW	$(0.9, 0.999)$
λ_P	20	λ_T	1
λ_R	20	λ_{S_T}	1
λ_{S_R}	20		

Table 5. **Simulator Configurations**. The most important parameters contain general settings (first two parameters) and material settings (the remaining). The parameters that differ from default simulator settings are **bolded**.

Parameter	Description	Value
`gravity`	The gravity force.	**0.0**
`substeps`	Substeps between trajectory waypoints.	60
`ka`	The contact adhesion distance.	0.0
`kd`	The contact damping stiffness.	1000.0
`ke`	The contact elastic stiffness.	100000.0
`kf`	The contact friction stiffness.	**0.0**
`mu`	The coefficient of friction.	**0.0**
`restitution`	The coefficient of restitution.	0.0
`thickness`	The thickness of the shape.	1e-05

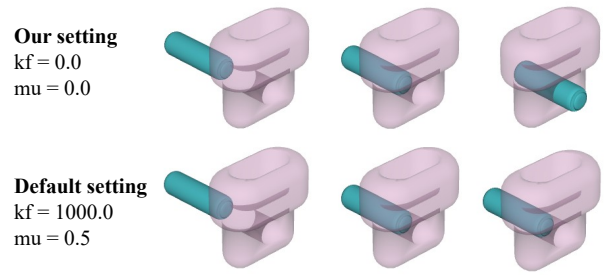


Figure 4. **Effects of Friction Parameters**. (Top) when disabling the friction effects in the simulator (our setting), the orange part can be successfully installed under the guidance of ground truth trajectory. (Bottom) default friction setting leads to a stuck at the rim of the hole.

4. Performance of Classic Motion Planning

We choose RRT and RRT-connect (used in [2]) to demonstrate classical motion planning methods are not suitable for the assembly trajectory generations in our scenario where the final poses are predicted. We conduct this experiment on a Windows Subsystem for Linux with an Intel Core i9-14900K CPU (32 cores) and 128 GB RAM.

In the following experiments, we feed the predicted final poses from AssemblyDyno to the two motion planning methods, instructing them to calculate the corresponding assembly motion. We inspect if they can provide solutions, no matter the quality, within given time constraints.

Table 6 highlights two major limitations of classical motion planners. First, these methods are inherently slow and struggle to find solutions under practical time constraints. Even with generous limits such as 30 s or 60 s, both RRT and RRT-Connect achieve success rates below 10%, indicating that they rarely return solutions quickly enough to be useful in real-world assembly scenarios.

More importantly, even when a very long time limit is imposed (120s), classical planners still fail to solve more than a small fraction of the tasks. This shows that classical planners require strictly collision-free goal states. However, the predicted final poses from AssemblyDyno may contain minor shape overlaps or small interpenetrations between parts, which are unavoidable when predictions are generated by learning-based models. These small inconsistencies cause classical planners to reject the goal configuration or become stuck while attempting to resolve infeasible collisions, preventing them from producing valid trajectories even with unlimited compute.

Assemble-them-all uses a search heuristic to produce part trajectories, whose computational complexity is combinatorial in the number of parts. While it takes 6.7s to produce assemblies on average, we found that tail cases take much longer, e.g., only 57.5% achieve success at $\leq 30s$.

Table 6. **Success rate of non-neural motion planning**. We present the success rate of returning answers (regardless of their quality) under varying time constraints. We use the predicted final poses from AssemblyDyno as inputs.

Algorithm	**Success Rate (%) with Timeout Constraints**						
	1s	2s	5s	10s	30s	60s	120s
RRT	2.9	3.6	4.6	5.0	6.4	6.8	7.1
RRT-Connect	2.9	3.2	3.6	4.6	5.4	6.1	7.1
Assemble Them All	4.6	8.6	21.8	34.6	57.5	72.1	81.8

Instead, our AssemblyDyno predicts all part trajectories *in a single forward pass*.

5. Qualitative Results

Figure 5 illustrates user-manual assembly instructions alongside our model's predicted trajectories. For each step, we visualize the motion as a sequence of temporally ordered point-cloud snapshots rendered as semi-transparent overlays. Although each trajectory contains 12 time steps, we display only the 1st, 6th, and 12th steps to provide a clear yet concise depiction of the object's motion during assembly. These overlaid time-step frames highlight how the predicted trajectories evolve over time and how the parts move toward their final configurations in each manual step.

6. Limitation

Unlike IKEA-style furniture parts, which often have significant part symmetries and duplications, the complex industrial parts in AssemblyBench (including large variation in part sizes) are found to be sensitive to even minor changes in the camera viewpoints. We believe new approaches for view-invariant diagram representations are necessary, and our multiview data generation pipeline facilitates research into this important topic.

7. Dataset Construction Details

7.1. User Study of the User Manuals

We conduct a two-stage user survey to evaluate the quality of our part names and text instructions. First, participants are shown 200 assembled CAD shapes, each with two rendered views and color-labeled parts, and are asked to count incorrectly named parts (*e.g.*, a cube labeled as "sphere"). As shown in Fig. 6, 72.9% of shapes contain zero incorrect names, and over 90% contain at most one or two, indicating high semantic accuracy.

Second, for each assembly we sample one instruction step and provide the current diagram, prior diagram, text instruction, and a labeled reference image. Participants rate text quality (1–10) and verify whether the spatial instruction

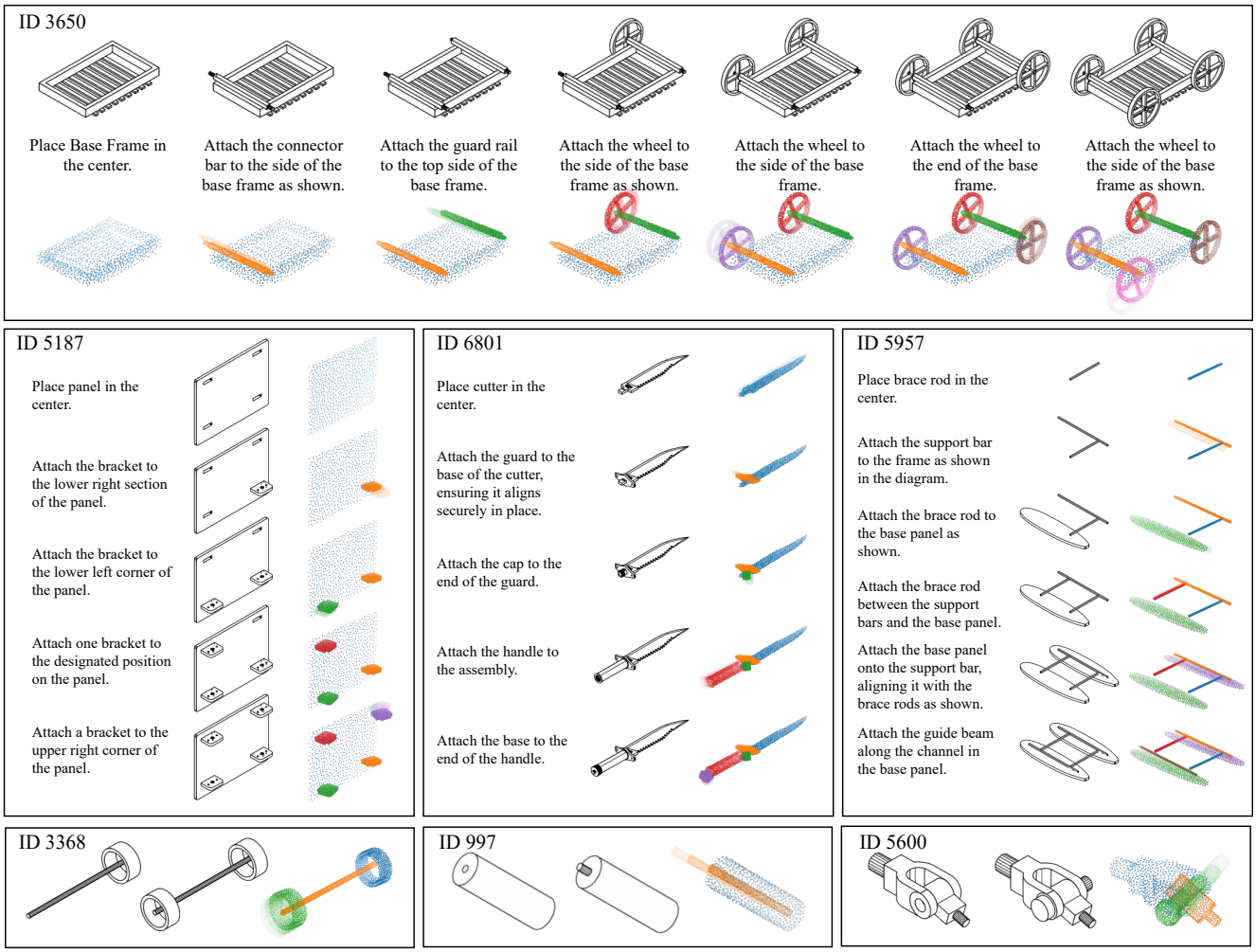


Figure 5. **User manuals with predicted trajectories**. We present the user manual and predicted trajectories of AssemblyDyno as the colored point clouds. Top two rows illustrate complete user manuals while the last row features insertion assembly steps. Multiple time steps are overlapped as transparent layers. The trajectories are executed in the stimulator, showing their outcomes when considering physical constraints.

matches the diagram. Results show that 54.1% of text instructions receive a rating of 10, and over two-thirds score 9 or above. Spatial correctness is also high: 91.4% of instructions are labeled correct. These outcomes confirm that our part names, textual descriptions, and spatial references are reliably annotated, providing a strong foundation for downstream tasks.

7.2. Choose Diagram Camera Views

As we render all diagrams in a set of camera views, for each shape assembly, We apply a heuristic to select the camera view that best demonstrate the assembly process. The objective is to choose, for each assembly part, a camera view that provides strong visual coverage of the part during the relevant assembly step and in the final assembled state. The method proceeds in three conceptual stages: (1) measur-

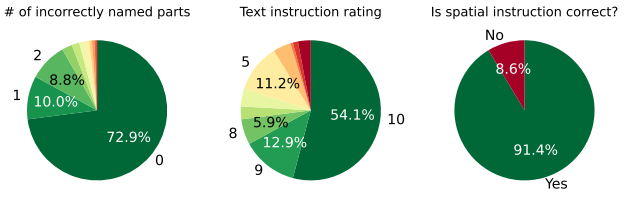


Figure 6. **User study on text annotation quality**. We present: (left) the distribution of incorrect part name amounts within a sampled assembly shape; (middle) the distribution of subjective text instruction ratings among all sampled shapes, higher is better; (right) the proportion of correct spatial information in the text instructions.

ing visibility, (2) scoring and normalizing per-part visibility, and (3) combining scores across the entire assembly to produce a consistent camera assignment.

Visibility Measurement Consider a set of cameras indexed by $c \in \mathcal{C}$, and a sequence of assembly parts indexed in order by $p \in \mathcal{P}$. For each camera c and part p, we observe two images:

- the diagram taken at the assembly step when part p is assembled, and
- the diagram taken at the final assembly completion.

From each diagram image we extract the number of pixels belonging to part p. Let $n_{c,p}^{(A)}$ and $n_{c,p}^{(F)}$ denote, respectively, the number of pixels of part p visible in camera c during the assembly step and during the final step. Thus, visibility is characterized by the pair $\left(n_{c,p}^{(A)},\ n_{c,p}^{(F)}\right)$.

Per-Part Visibility Score Visibility should contribute to the score in a way that has diminishing returns for large pixel counts. A logarithmic visibility score fulfills this requirement. For each camera c and part p, define:

$$s_{c,p} = \begin{cases} \log\left(1 + \lambda\, n_{c,p}^{(A)}\right) + \log\left(1 + \lambda\, n_{c,p}^{(F)}\right), & \text{if } n_{c,p}^{(A)} > 0, \\ 0, & \text{if } n_{c,p}^{(A)} = 0, \end{cases}$$

where λ is a scaling constant set as 0.05 that moderates the influence of raw pixel counts.

This construction ensures:

- both assembly-step visibility and final-step visibility contribute additively,
- visibility in the assembly step is essential (otherwise the score is zero),
- visibility contributions grow sublinearly.

For each part p, the visibility scores $\{s_{c,p}\}_{c\in\mathcal{C}}$ are divided by the max score across all cameras ($\max_{c\in\mathcal{C}} s_{c,p}$), so that the best camera attains a normalized score $\hat{s}_{c,p}$.

Aggregated Camera Quality Across All Parts To determine which cameras are most useful across the entire assembly process, the normalized per-part scores are summed over all parts to S_c.

$$S_c = \sum_{p\in\mathcal{P}} \hat{s}_{c,p}.$$

The quantity S_c expresses the overall usefulness of camera c across the entire assembly. The cameras are ranked in descending order of S_c. This global ranking reflects which cameras tend to provide good visibility for many parts.

7.3. Instructional Text Generation

We use VLMs to name the CAD parts one by one. For each CAD part, we provide the VLM with the following text prompts and three images:

- The diagram from the first step where the part is introduced, with the part highlighted.
- The final-step diagram of the completed assembly, also highlighting the same part.
- When there are similar parts in the assembly, we provide an additional diagram where all its counterparts are highlighted.

> You are a product design assistant who write user manuals.
> You should name the colored component in the first image.
> In the second image, you are given the final assembled model with your focus component colored.
> In the third image, all similar components are colored for your reference. You should give a general name (in singular form) for all these similar components.
>
> – Adopt one name. Don't include multiple choices.
> – Don't include color into names.
> – Avoid using oriental words such as left, right, horizontal.
> – Avoid existing part names: {existing_names}
>
> Think in steps and finalize your answer in json.
> Example output:
>
> ```
> json
> {
>   "name": "Feet"
> }
>
> json
> {
>   "name": "Connector Bar"
> }
> ```

We find similar parts by grouping their bounding box sizes. As we prompt the VLM to assign a general name for each group, we only name one part within the group.

For all diagrams, we choose the best camera view from a predefined set, by colorizing the target part and selecting the view that maximizes the number of colored pixels of our target part in the diagram image. The camera views vary among CAD parts, which is different from final user manuals, where all diagrams share the same view within the shape assembly.

We use the generated names to create text instructions for final user manuals. We generate assembly text instructions step-by-step. Within each VLM call, we provide the following text prompts with two images, where the camera views align with the final user manuals.:

- The diagram of the current assembly step, with the part to be assembled highlighted in color.
- The same diagram of the same step, but with all of the parts highlighted in varying colors and labeled by their names.

You are a product design assistant who write user manuals.
You should describe the assembly step of the first image, which involves only one component highlighted in color.
The names of the current component and previous components are shown in the second image as a reference.

– Just describe the current assembly step.
– Don't include instructions for previous steps and components.
– Correct the incorrect plural form of the component name if any.
– Don't subdivide the current step into multiple sub–steps.
– Avoid words like "as shown in the image" or "as illustrated".
– Don't include color in your description.

Think in steps and finalize your answer in json.
Example output:

```
json
{
  "text": "your description here"
}
```

References

[1] Newton Contributors. Newton: GPU-accelerated physics simulation for robotics, and simulation research., 2025. 3

[2] Chenrui Tie, Shengxiang Sun, Jinxuan Zhu, Yiwei Liu, Jingxiang Guo, Yue Hu, Haonan Chen, Junting Chen, Ruihai Wu, and Lin Shao. Manual2skill: Learning to read manuals and acquire robotic skills for furniture assembly using vision-language models. *arXiv preprint arXiv:2502.10090*, 2025. 4

[3] Jiahao Zhang, Anoop Cherian, Cristian Rodriguez, Weijian Deng, and Stephen Gould. Manual-PA: Learning 3d part assembly from instruction diagrams. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6304–6314, 2025. 2